

# MOSAIC: A Proximity Graph Approach for Agglomerative Clustering

Jiyeon Choo<sup>1</sup>, Rachsuda Jiamthaphaksin<sup>1</sup>, Chun-sheng Chen<sup>1</sup>,  
Oner Ulvi Celepcikay<sup>1</sup>, Christian Giusti<sup>2</sup>, and Christoph F. Eick<sup>1</sup>

<sup>1</sup>Computer Science Department, University of Houston, Houston, TX 77204-3010, USA

<sup>2</sup>Department of Mathematics and Computer Science, University of Udine, Via delle Scienze,  
33100, Udine, Italy

<sup>1</sup>{jchoo, rachsuda, lyons19, onerulvi, ceick}@cs.uh.edu, <sup>2</sup>giusti@dimi.uniud.it

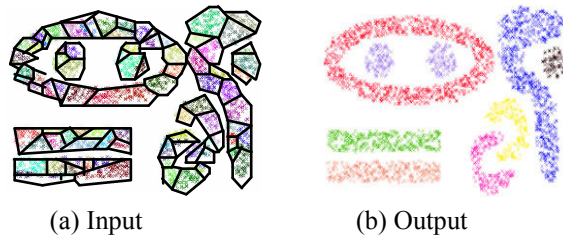
**Abstract.** Representative-based clustering algorithms are quite popular due to their relative high speed and because of their sound theoretical foundation. On the other hand, the clusters they can obtain are limited to convex shapes and clustering results are also highly sensitive to initializations. In this paper, a novel agglomerative clustering algorithm called MOSAIC is proposed which greedily merges neighboring clusters maximizing a given fitness function. MOSAIC uses Gabriel graphs to determine which clusters are neighboring and approximates non-convex shapes as the unions of small clusters that have been computed using a representative-based clustering algorithm. The experimental results show that this technique leads to clusters of higher quality compared to running a representative clustering algorithm stand-alone. Given a suitable fitness function, MOSAIC is able to detect arbitrary shape clusters. In addition, MOSAIC is capable of dealing with high dimensional data.

**Keywords:** Post-processing, hybrid clustering, finding clusters of arbitrary shape, agglomerative clustering, using proximity graphs for clustering.

## 1 Introduction

Representative-based clustering algorithms form clusters by assigning objects to the closest cluster representative.  $k$ -means is the most popular representative-based clustering algorithm: it uses cluster centroids as representatives and iteratively updates clusters and centroids until no change in the clustering occurs.  $k$ -means is a relatively fast clustering algorithm with a complexity of  $O(ktn)$ , where  $n$  is the number of objects,  $k$  is the number of clusters, and  $t$  is the number of iterations. The clusters generated are always contiguous. However, when using  $k$ -means the number of clusters  $k$  has to be known in advance, and  $k$ -means is very sensitive to initializations and outliers. Another problem of  $k$ -means clustering algorithm is that it cannot obtain clusters that have non-convex shapes [1]: the shapes that can be obtained by representative-based clustering algorithms are limited to convex polygons.

In theory, agglomerative hierarchical clustering (AHC) [2] is capable of detecting clusters of arbitrary shape. However, in practice, it performs a very narrow search, merging the two closest clusters without considering other merging candidates and therefore often misses high quality solutions. Moreover, its time complexity is  $O(n^2)$  or worse. Finally, many variations of AHC obtain non-contiguous clusters. [3]



**Fig. 1.** An illustration of MOSAIC's approach

This paper proposes a hybrid clustering technique that combines representative-based with agglomerative clustering trying to maximize the strong points of each approach. A novel agglomerative clustering algorithm called MOSAIC is proposed, which greedily merges neighboring clusters maximizing a given fitness function and whose implementation uses Gabriel graphs [4] to determine which clusters are neighboring. Non-convex shapes are approximated as the union of small convex clusters that have been obtained by running a representative-based clustering algorithm, as illustrated in Fig. 1. Creating mosaics in art is the process of assembling small pieces to get a sophisticated design. Similarly, the proposed MOSAIC algorithm pieces convex polygons together to obtain better clusters.

```

1. Run a representative-based clustering algorithm to create a
   large number of clusters.
2. Read the representatives of the obtained clusters.
3. Create a merge candidate relation using proximity graphs.
4. WHILE there are merge-candidates  $(C_i, C_j)$  left
   BEGIN
       Merge the pair of merge-candidates  $(C_i, C_j)$ , that
       enhances fitness function  $q$  the most, into a new
       cluster  $C'$ 
       Update merge-candidates:
        $\forall C$  Merge-Candidate( $C', C$ )  $\Leftrightarrow$ 
           Merge-Candidate( $C_i, C$ )  $\vee$  Merge-Candidate( $C_j, C$ )
   END
RETURN the best clustering  $X$  found.

```

**Fig. 2.** Pseudo code for MOSAIC

Relying on proximity graphs the MOSAIC conducts a much wider search which, we claim, results in clusters of higher quality. Moreover, the expensive, agglomerative clustering algorithm is only run for usually less than 1000 iterations;

therefore, the impact of its high complexity on the overall run time is alleviated, particularly for very large data sets. Furthermore, the proposed post-processing technique is highly generic in that it can be used with any representative-based clustering algorithm, with any proximity graph and with any cluster evaluation function. Fig. 2 gives the pseudo code of the proposed MOSAIC algorithm.

In summary, MOSAIC merges pairs of neighboring clusters maximizing an externally given fitness function  $q$ , and this process is continued until only one cluster is left. Finally, the best clustering is determined and returned. Using cluster representatives obtained from a representative-based clustering algorithm as an input, a proximity graph is generated to determine which of the original clusters are neighboring and a merge-candidate relation is constructed from this proximity graph. When clusters are merged, this merge-candidate relation is updated incrementally without any need to regenerate proximity graphs.

The main contributions of the paper are;

- It introduces a hybrid algorithm that combines strong features of representative-based clustering and agglomerative clustering.
- The algorithm provides flexibility by enabling to plug-in any fitness functions and is not restricted to any specific cluster evaluation measure.
- The algorithm conducts a much wider search, compared to traditional agglomerative clustering algorithms, by considering neighboring clusters as merge candidates.

The organization of our paper is as follows. Section 2 describes MOSAIC in more detail. Then the performance study of MOSAIC and the comparative study with DBSCAN and  $k$ -means are explained in section 3. Related work is reviewed in Section 4, and a conclusion is given in Section 5 respectively.

## 2 Post-Processing with MOSAIC

This section discusses MOSAIC in more detail. First, proximity graphs are introduced and their role in agglomerative clustering is discussed. Next, an internal cluster evaluation measure will be discussed that will serve as a fitness function in the experimental evaluation. Finally, MOSAIC's complexity is discussed

### 2.1 Using Gabriel Graphs for Determining Neighboring Clusters

Different proximity graphs represent different neighbor relationships for a set of objects. There are various kinds of proximity graphs [5], with Delaunay graphs [6] (DG) being the most popular ones. The Delaunay graph for a set of cluster representatives tells us which clusters of a representative-based clustering are neighboring and the shapes of representative-based clusters are limited to Voronoi cells, the dual to Delaunay graphs.

Delaunay triangulation (DT) [7] is the algorithm that constructs the Delaunay graphs for a set of objects. Unfortunately, using DT for high dimensional datasets is

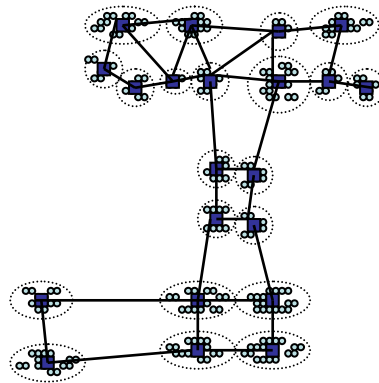
impractical since it has a high complexity of  $O(n^{\frac{d}{2}})$  (when  $d > 2$ ), where  $d$  is the number of dimensions of a data set. Therefore, our implementation of MOSAIC uses another proximity graph called Gabriel graphs (GG) [4] instead, which is a sub-graph of the DG. Two points are said to be Gabriel neighbors if their diametric sphere does not contain any other points. The pseudo code of an algorithm that constructs the GG for a given set of objects is given in Fig. 3. Constructing GG has a time complexity of  $O(dn^3)$  but faster, approximate algorithms ( $O(dn^2)$ ) to construct GG exist [8].

Let  $R = \{r_1, r_2, \dots, r_k\}$ , be a set of cluster representatives  
 FOR each pair of representatives  $(r_i, r_j)$ ,  
 IF for each representative  $r_p$ , the following inequality  

$$d(r_i, r_j) \leq \sqrt{d^2(r_i, r_p) + d^2(r_j, r_p)}$$
 where  $p \neq i, j$  and  $r_p \in R$ , is true,  
 THEN  $r_i$  and  $r_j$  are neighboring.  
 $d(r_i, r_j)$  denotes the distance of representatives  $r_i$  and  $r_j$ .

**Fig. 3.** Pseudo code for constructing Gabriel graphs.

Gabriel graphs are known to provide good approximations of Delaunay graphs because a very high percentage of the edges of a Delaunay graph are preserved in the corresponding Gabriel graph [9]. MOSAIC constructs the Gabriel graph for a given set of representatives, e.g. cluster centroids in the case of  $k$ -means, and then uses the Gabriel graph to construct a boolean merge-candidate relation that describes which of the initial clusters are neighboring. This merge candidate relation is then updated incrementally when clusters are merged. The illustration of the Gabriel graph construction in MOSAIC is shown in Fig. 4 in which cluster representatives are depicted as squares, objects in a cluster are represented using small blue circles, and clusters that have been obtained by a representative algorithm are visualized using dotted lines.



**Fig. 4.** Gabriel graph for clusters generated by a representative-based clustering algorithm

## 2.2 Cluster Evaluation Measures for Traditional Clustering

Many evaluation measures have been proposed in the literature [2]. In this paper, we use Silhouettes [10] which is an internal evaluation measure that has been widely used to assess cluster quality. Silhouettes is a popular cluster evaluation technique that takes into consideration both cohesion and separation. The definition of Silhouettes is as follows:

$$s_i = \frac{(a_i - b_i)}{\max(a_i, b_i)} \quad (1)$$

where,

$$a_i = \min_m \left( \frac{1}{|C_m|} \sum_{o_j \in C_m} d_{ij} \right), \quad m \neq k \quad \text{and} \quad (2)$$

$$b_i = \frac{1}{|C_k|} \sum_{o_j \in C_k} d_{ij}. \quad (3)$$

In the formula (1),  $b_i$  is average dissimilarity of an object  $o_i$  to all other objects  $o_j$  in the same cluster.  $a_i$  is minimum of average dissimilarity of an object  $o_i$  to all objects  $o_j$  in another cluster (the closest cluster). To measure quality not for one object but entire clustering, we use average of Silhouettes over whole dataset. The fitness function  $q(X)$  is defined as follows:

$$q(X) = \frac{1}{n} \sum_{i=1}^n s_i \quad (4)$$

where  $n$  is the number of objects in a dataset.

## 2.3 Complexity

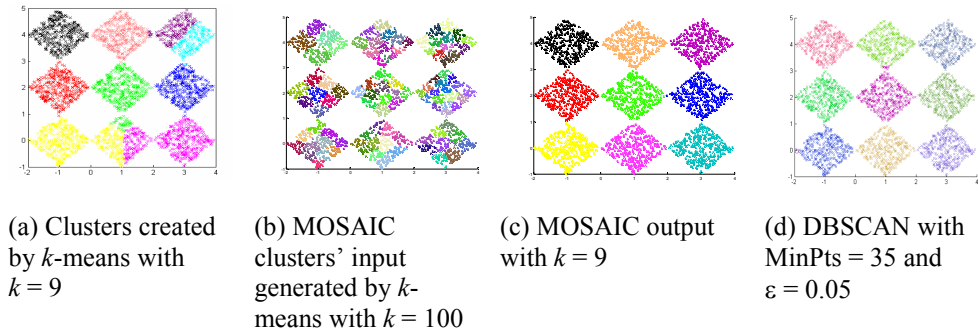
The time complexity of our proposed hybrid clustering algorithm depends on two factors: the complexity of the representative-based clustering algorithm and of the MOSAIC algorithm itself. Analyzing MOSAIC's complexity, we already discussed that the cost for constructing the Gabriel Graph is  $O(k^3)$ . After that, we have to merge the  $k$  vertices of the Gabriel Graph. Basically, a Delaunay Graph is a planar graph; since a Gabriel Graph is a connected subset of a Delaunay Graph, we have that the number  $e$  of edges of our GG is  $k-1 \leq e \leq 3k-6$ . This means that the number of edges  $e$  in the graph is always linear with respect to the number of vertices:  $e=O(k)$ . Thus, at the  $i^{\text{th}}$  iteration,  $O(k-i)$  new merge-candidates are created for the newly created cluster that have to be evaluated, which adds up to  $O(k^2)$  fitness function evaluations:  $(O(k-1) + O(k-2) + \dots + 1)$ . Putting this all together, we obtain a time complexity of the MOSAIC algorithm of:  $O(k^3 + k^2 * (O(q(X))))$  where  $O(q(X))$  is the time complexity

of the fitness function. A lower complexity for MOSAIC can be obtained if the fitness of a particular clustering can be computed incrementally during the merging stages based on results of previous fitness computations.

### 3 Experiments

We compare MOSAIC using the Silhouettes fitness function with DBSCAN and  $k$ -means<sup>1</sup>. Due to space limitations we are only able to present a few results; a more detailed experimental evaluation can be found in [3]. We conducted our experiments on a Dell Inspiron 600m laptop with a Intel(R) Pentium(R) M 1.6GHz processor with 512 MB of RAM. We set up three experiments that use the following datasets: an artificial dataset called **9Diamonds**[11] consisting of 3,000 objects with 9 natural clusters, **Volcano**[11] containing 1,533 objects, **Diabetes**[12] containing 768 objects, **Ionosphere**[12] containing 351 objects, and **Vehicle**[12] containing 8,469 objects.

**Experiment 1:** The experiment compares the clustering results generated by running  $k$ -means with  $k=9$  with MOSAIC for the 9Diamonds dataset.



**Fig. 5.** Experimental results for the 9Diamonds dataset

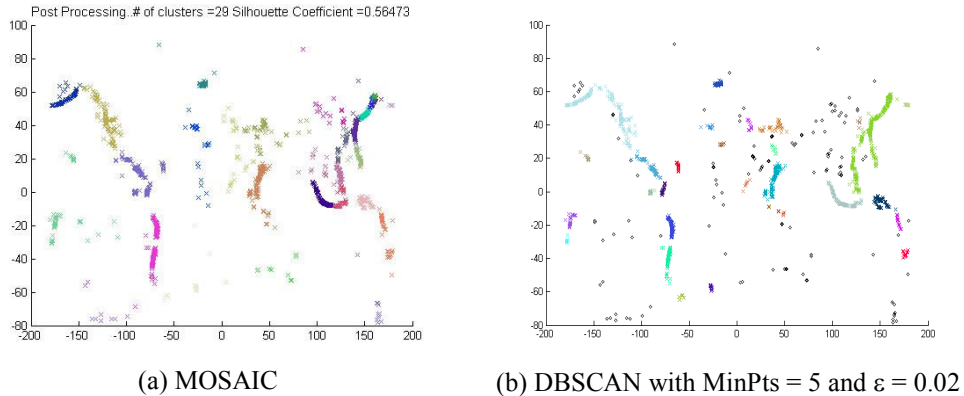
**Discussion:** As shown in Fig. 5 (a),  $k$ -means is not able to discover the natural clusters. MOSAIC, on the other hand, is able to discover the natural clusters by iteratively merging the sub-clusters that have been depicted in Fig. 5 (b) by maximizing the Silhouettes fitness function: the clustering with the highest fitness value is displayed in Fig. 5 (c).

**Experiment 2:** The experiment compares the clustering results generated by MOSAIC and DBSCAN for two two-dimensional datasets: 9Diamonds and Volcano.

<sup>1</sup> In general, we would have preferred to compare our algorithm also with CHAMELEON and DENCLUE. However, we sadly have to report that executable versions of these two algorithms no longer exist.

**Discussion:** To use DBSCAN, we have to choose values for two parameters: MinPts and  $\epsilon$ . One challenge of this experiment is to find proper values for those parameters. First, we used the procedure that has been proposed in the original paper [13] to select values for MinPts and  $\epsilon$ . Unfortunately, this procedure did not work very well: DBSCAN just created a single cluster for both datasets tested. Therefore, relying on human intuition, we employed a manual, interactive procedure that generated 80 pairs of parameters for DBSCAN. The parameters selected by the second procedure lead to much better results. We observed that  $\epsilon$  values that produce better clustering results are much smaller than those suggested by analyzing the sorted  $k$ -dist graph. Fig. 5 (d) depicts one of the best clustering results obtained for DBSCAN for the 9Diamonds dataset. MOSAIC correctly clusters the dataset while DBSCAN reports a small number of outliers in the left corner of the bottom left cluster.

Volcano is a real world dataset that contains chain-like patterns with various densities. In general, DBSCAN and MOSAIC produced results of similar quality for this dataset. Fig. 6 depicts a typical result of this comparison: MOSAIC does a better job in identifying the long chains in the left half of the display (Fig. 6 (a)), whereas DBSCAN correctly identifies the long chain in the upper right of the display (Fig. 6. (b)). DBSCAN and MOSAIC both fail to identify all chain patterns.



**Fig. 6.** Experimental results of MOSAIC and DBSCAN on Volcano dataset

**Experiment 3:** This experiment compares MOSAIC and  $k$ -means on three high dimensional datasets: Vehicle, Ionosphere, and Diabetes. The quality of clustering results is compared using the Silhouettes cluster evaluation measure. MOSAIC's input were 100 clusters that have been created by running  $k$ -means. Next, MOSAIC was run and its Silhouette values were averaged over its 98 iterations. These Silhouette values were compared with the average Silhouette values obtained by running  $k$ -means with  $k = 2 - 99$ . Table 1 summarizes the findings of that experiment.

**Table 1.** Information for the high dimensional datasets and experimental results

Dataset	Number of objects	Number of dimensions	Average Silhouette coefficient of $k$ -means	Average Silhouette coefficient of MOSAIC
Vehicle	8,469	19	0.20013	0.37157
Ionosphere	351	34	0.2395	0.26899
Diabetes	768	8	0.23357	0.24373

**Discussion:** MOSAIC outperforms  $k$ -means quite significantly for the Vehicle dataset and we see minor improvements for the Ionosphere and Diabetes datasets.

## 4 Related Work

Discovering arbitrary shape clusters is very important in many domains such as hot spot detection, region discovery and spatial data mining. Jiang [1] proposes spatial clustering techniques that employ hierarchical clustering accompanied by tree-like diagrams and claim that this is a beneficiary for visualizing cluster hierarchies at different levels of detail. Anders [14] developed an unsupervised graph-based clustering algorithm, called Hierarchical Parameter-free Graph Clustering (HPGCL) for spatial data analysis. Various proximity graphs are used to define coarse-to-fine hierarchical segmentation of data.

Density-based clustering methods [13, 15, 16, and 17] have been found to be efficient for discovering dense arbitrary shaped clusters, such as the ones in the 9Diamonds dataset. The main drawbacks of density-based clustering algorithms are their need for parameters tuning, and their usually poor performance for datasets with varying density. Moreover, they do not seem to be suitable for high dimensional data.

There has been significant research centering on hybrid clustering. CURE is a hybrid clustering algorithm that integrates a partitioning algorithm with an agglomerative hierarchical algorithm [18]. CURE agglomeratively merges the two clusters that have the closest pair of representatives, and updates mean and a set of representative points. CHAMELEON [19] provides a sophisticated two-phased clustering algorithm. In the first phase, it uses a multilevel graph partitioning algorithm to create an initial set of clusters and in the second phase it iteratively merges clusters maximizing relative inter-connectivity and relative closeness. MOSAIC also relies on two-phase clustering but it has a major advantage over CHAMELEON and CURE by being able to plug-in any fitness function and not being restricted to evaluate clusters based on inter-connectivity and closeness. Lin and Zhong [20 and 21] propose hybrid clustering algorithms that combine representative-based clustering and agglomerative clustering methods. However they employ different merging criteria and perform a narrow search that only considers a single pair of merge candidates. Surdeanu [22] proposes a hybrid clustering approach that combines agglomerative clustering algorithm with the Expectation Maximization (EM) algorithm.

## 5 Conclusion

This paper proposes a novel approach that approximates arbitrary-shape clusters through unions of small convex polygons that have been obtained by running a representative-based clustering algorithm. An agglomerative clustering algorithm called MOSAIC is introduced that greedily merges neighboring clusters maximizing an externally given fitness function. Gabriel graphs are used to determine which clusters are neighboring. We claim that using proximity graphs increases the number of merge candidates considerably over traditional agglomerative clustering algorithms that only consider “closest” clusters for merging, resulting in clusters of higher quality. MOSAIC is quite general and can be used with any representative-based clustering algorithm, any proximity graph, and any fitness function. Moreover, we claim that MOSAIC can be effectively applied to higher dimensional data.

MOSAIC also has some similarity with agglomerative grid-based clustering algorithms; both approaches employ micro-clusters which are grid-cells in their approach and convex polygons in our approach and greedily merge neighboring clusters. However, our approach is much more general by supporting more variety of shapes and it allows for convex polygons of different sizes. On the other hand, for a given grid structure it is easy to determine which clusters are neighboring, which is not true for our approach.

We conducted experiments whose results suggest that using MOSAIC in conjunction with  $k$ -means can significantly improve cluster quality. Using Silhouettes function as a fitness function we also compared MOSAIC with DBSCAN; both algorithms obtained results of similar quality for most datasets tested. However, before using DBSCAN we had to spend significant efforts for parameter tuning which is not the case when using MOSAIC which only requires a single input parameter: the fitness function.

However, based on our initial experimental results, we don't believe that the Silhouettes function is the best possible fitness function to find arbitrary shape clusters. Consequently, in our current research we investigate to find more suitable fitness functions for this purpose.

## References

1. Jiang, B., Spatial Clustering for Mining Knowledge in Support of Generalization Processes in GIS. In ICA Workshop on Generalisation and Multiple representation (2004)
2. Tan, M., Steinbach, M., Kumar, V., Introduction to Data Mining, 1st edn. Addison Wesley (2005)
3. Choo, J., Using Proximity Graphs to Enhance Representative-based Clustering Algorithms. Master Thesis, Department of Computer Science, University of Houston, TX, (2007)
4. Gabriel, K., R. Sokal, A New Statistical Approach to Geographic Variation Analysis. In Systematic Zoology, Vol. 18. (1969) 259-278
5. Toussaint, G., The Relative Neighborhood Graph of A Finite Planar Set. Int. Conf. Pattern Recognition, Vol. 12. (1980) 261-268
6. Kirkpatrick, D., A note on Delaunay and Optimal Triangulations. In Information Processing Letters 10, (1980) 127-128

7. Okabe, A., Boots, B., Sugihara, K., Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. Wiley, New York (1992)
8. Bhattacharya, B., Poulsen, R., Toussaint, G., Application of Proximity Graphs to Editing Nearest Neighbor Decision Rule. Int. Sym. on Information Theory (1981)
9. Asano, T., Ibaraki, T., Imai, H., and Nishizeki, T., Algorithms. Lecture Notes in Computer Science, Springer-Verlag, New York Berlin Heidelberg (1990) 70-71
10. Rousseeuw, P.J., Silhouettes: A Graphical Aid to The Interpretation and Validation of Cluster Analysis. Int. J. Computational and Applied Mathematics 20. (1987) 53-65
11. Data Mining and Machine Learning Group website, University of Houston, Texas, <http://www.tlc2.uh.edu/dmmlg/Datasets>
12. UCI Machine Learning Repository, <http://www.ics.uci.edu/~mllearn/MLRepository.html>
13. Ester, M., Kriegel, H.P., Sander, J., Xu, X., Density-Based Spatial Clustering of Applications with Noise. Int. Conf. Knowledge Discovery and Data Mining (1996)
14. Anders, K.H., A Hierarchical Graph-Clustering Approach to Find Groups of Objects. Technical Paper, In ICA Commission on Map Generalization, 5th Workshop on Progress in Automated Map Generalization. (2003)
15. Sander, J., Ester, M., Kriegel, H.P. Xu, X., Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. Inf. Conf. Data Mining and Knowledge Discovery. (1998) 169-194
16. Kriegel, H.P. Pfeifle, M., Density-Based Clustering of Uncertain Data. In Int. Conf. Knowledge Discovery in Data Mining (2005) 672-677
17. Hinneburg, A., Keim, D., An Efficient Approach to Clustering in Large Multimedia Databases with Noise. Conf. Knowledge Discovery in Data Mining, 1998
18. Guha, S., Rastogi, R., Shim, K., CURE: An Efficient Clustering Algorithm for Large Databases. Int. Conf. ACM SIGMOD on Management of data. (1998) 73-84
19. Karypis, G., Han, E.H., Kumar, V., CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. In IEEE Computer Vol. 32 (1999) 68-75
20. Lin, C., Chen, M., A Robust and Efficient Clustering Algorithm based on Cohesion Self-Merging. Inf. Conf. 8th ACM SIGKDD on Knowledge Discovery and Data Mining. (2002) 582-587
21. Zhong, S., Ghosh, J., A Unified Framework for Model-based Clustering. Int. J. Machine Learning Research Vol.4.(2003) 1001-1037
22. Surdeanu, M., Turmo, J. Ageno, A., A Hybrid Unsupervised Approach for Document Clustering. Int. Conf. 11h ACM SIGKDD on Knowledge Discovery in Data Mining (2005) 685-690